

Дисципліна “Об’єктно-орієнтоване програмування”

Структура дисципліни

Аудиторні заняття:

- | | |
|---------------------|---------|
| - Лекції | 8 год. |
| - Практичні заняття | 24 год. |

Самостійна робота
Форма контролю

73 год.
ПМК

Викладач Чибіряк Яна Іванівна
Кафедра ІКТ (секція ІТП)
Ауд. Г1306

№ п/п	Назва навчально - методичних матеріалів	Вид	Наявність примірн
1. Навчальна література (підручники, навчальні посібники)			
1	Фридман А.Л. Основы объектно-ориентированного программирования на языке С++. – М.: Горячая линия – Телеком, Радио и связь, 1999. – 208 с.	Книга	45
2	Шилдт Г. С++: руководство для начинающих. – М.: Изд. Дом. «Вильямс», 2005. – 672 с.	Навч. посіб.	1 (ел. вигляд.)
3	Баженова И.Ю. С++ & Visual studio.NET. Самоучитель программиста. – М.: КУДИЦ-ОБРАЗ, 2003. – 448 с.	Ел. докум.	-
2. Навчально - методичні матеріали для забезпечення практичних занять			
4	Методичні вказівки до виконання лабораторних робіт із дисципліни «Системне програмування та операційні системи» / укладачі: С. М. Ващенко, Я. І. Чибіряк.	Друк.	20

Регламент

Розподіл рейтингових балів :

- **Робота на аудиторних заняттях** ($0,16 \times R = 16$ балів):
- Лекції: 4 лк. \times 1 бал/лк. = 4 балів.
- Лабораторні заняття: 12 лб. \times 1 бал/пр. = 12 балів.

Виконання практичних робіт (12 завдань) – максимально 60 балів (при позитивному оцінюванні з кожної практичної роботи до 5 балів);;

- **Складання модульних тестових контролів** – всього 24 бали (один у кожному модульному циклі).
- **Шкала оцінювання**: у першому модульному циклі – 12 балів, у другому – 12 балів.

Підсумок рейтингових балів за мод. циклами :

- 1-ий модульний цикл: (4 лк., 4 пр. робіт, захист; модульний контроль) – до 40 балів.
- 2-ий модульний цикл: (8 пр. робіт, захист; модульний контроль) – до 60 балів.

ОСНОВНІ ВИЗНАЧЕННЯ МОВИ C++

Алфавіт мови - властивий цій мові набір символів, з яких формуються усі конструкції мови.

Для ідентифікації об'єктів програми (змінні, константи, типи, підпрограми та ін.) використовуються ідентифікатори, або імена.

Ідентифікатори починаються з латинської букви і можуть містити латинські букви, цифри і знаки підкреслення.

Ключові (службові, зарезервовані) слова мають однозначний зміст і можуть використовуватися тільки так, як це задано в даній мові програмування.

Алфавіт мови C (C++) включає:

- прописні латинські букви A .. Z;
- малі латинські букви a .. z;
- арабські цифри 0 .. 9;
- роздільники: ,, ,; , ? , ' , ! , | , / , \ , ~ , _ , # , % , & , ^ , = , - + * (,) { } [] < , > ;
- пробільні символи: SP, H_TAB, CR, LF, V_TAB, FF, Ctrl - Z (кінець текстового файлу);

ОСНОВНІ ВИЗНАЧЕННЯ МОВИ C++

- спеціальні символи:

Символ	Опис
\n	Новий рядок (LF)
\t	Горизонтальна табуляція (H_TAB)
\r	Повернення каретки (CR)
\f	Нова сторінка (FF)
\a	Звуковий сигнал
\'	Апостроф
\”	Лапки
\\	Обернена коса риска (back slash)

Директива препроцесора - це інструкція, що записується на початку програми, починається з символу «#» і виконується препроцесором.

Препроцесор переглядає програму до компіляції, підключає необхідні файли, замінює символічні аббревіатури в програмі на відповідні директиви і т. д.

ОСНОВНІ ВИЗНАЧЕННЯ МОВИ C++

- // Однорядковий коментар в C++
- /* Багаторядковий
- коментар в C++*/

Структура програми мовою C++

- `#include "stdafx.h"` // директиви препроцесора
- `#include <iostream>`
- `#include <conio.h>`
- `using namespace std;` // директива using
- `int _tmain(int arge, TCHAR* argv[])` // початок програми
- `{` // початок тіла програми
- `cout<<" Hello!\n"` // оператор виводу на екран
- `return 0;`
- `}` // кінець тіла програми

- **Приклад 1. Програма, що містить елементи обчислень.**
- `#include "stdafx.h"`
- `#include <iostream>`
- `#include <conio.h>`
- `using namespace std;`
- `int _tmain(int argc, _TCHAR* argv[])`
- `{ int a, b, c; // об'явлення змінних цілого типу`
`a = 5; b = 10; // визначення змінних a і b`
`c = a+b; // присвоєння змінній c суми a і b`
- `cout << "Value c = "<<c<<"\n";`
- `_getch(); // підключається за допомогою заголовного`
`// файлу conio.h,`
- `return 0; // зупиняє виконання програми до натиснення`
`// будь-якої клавіші`
- `}`
- *Маніпулятор endl* переводить курсор на новий рядок, виконує очищення буфера рядка.

- **Приклад 2.**
- `#include "stdafx.h"`
- `#include <iostream>`
- `#include <conio.h>`
- `using namespace std;`
- `int _tmain(int arge, _TCHAR* argv[])`
- `{ double m, p, q;`
- `cout<<"Input p and q :";`
- `cin>>p>>q;`
- `m=p/q;`
- `cout<<"m = "<<m<<endl;`
- `_getch();`
- `return 0; }`

Висновки:

- будь-яка програма містить заголовні файли, підключені директивою `#include`;
- програма завжди починається словом `main()`;
- перед використанням змінної її необхідно об'явити і вказати її тип.

- ***Чи усе вірно в тексті програмі?***

- # include <iostream>
- using namespace std;
- int main()
- { int m, k;
- m = k+2;
- cout<<m; }

- ***Виправити помилки, якщо вони є.***

- # include <iostream>
- using namespace std;
- int main()
- {float r;
- cout<<"Введіть значення a";
- cin>>a;
- cout<<"Введіть значення b";
- cin>>b;
- r=a*b;
- cout<<"r = "<<r<<endl;
- return 0; }

- **Що буде виведено на екран в результаті роботи програми, якщо $m=13$ і $l=4$?**

- `# include <iostream>`
- `using namespace std;`
- `int main()`
- `{ int p, l, m;`
- `cout<<" Введіть значення m";`
- `cin>>m;`
- `cout <<" Введіть значення l";`
- `cin>>l;`
- `l = 2; m = 9;`
- `p = m + l;`
- `cout<<"p = "<<p<<endl;`
- `return 0; }`

СТАНДАРТНІ ТИПИ ДАНИХ

- Ім'я змінної пов'язане з областю пам'яті, яка відведена для зберігання значення цієї змінної.
 - **Змінні цілого типу**

<i>Тип даних</i>	<i>Розмір, байт</i>	<i>Нижня межа діапазону</i>	<i>Верхня межа діапазону</i>
[signed] char	1	-128	127
unsigned char	1	0	255
[signed] short	2	-32 768	32 767
unsigned short	2	0	65 535
[signed] int	2	-32 768	32 767
	4	-2 147 483 648	2 147 483 647
unsigned int	2	0	65 535
	4	0	4 294 967 295
[signed] long	4	-2 147 483 648	2 147 483 647
unsigned long	4	0	4 294 967 295

СТАНДАРТНІ ТИПИ ДАНИХ

● Дійсні типи даних

<i>Тип даних</i>	<i>Розмір, байт</i>	<i>Точність</i>	<i>Нижня межа діапазону</i>	<i>Верхня межа діапазону</i>
float	4	1	3.4E-38	3.4E+38
double	8	15	1.7E-308	1.7E+308
long double	10	15	3.4E-4932	1.1E+4932

● Змінні логічного типу

- Тип **bool** може мати два значення: false (лож) і true (істина). Ці змінні фактично займають 1 біт, але транслятори виділяють під них по 1 байту пам'яті.

● Константи

- `const int intVal = 25;`
- `const float floatVal = 2.531;`
- `const char sym = '*';`

- **Приклад .**
- `const long int LI = 1257L;` // зберігається як long int
- `const unsigned int UI = 5317U;` // зберігається як unsigned int
- `const unsigned long int ULI = 2UL;` // зберігається як unsigned long int
- `const float PI = 3.14159F;` // зберігається як float
- `const float PI = 3.14159f;` // зберігається як float
- `const long double M = 2.58316L;` // зберігається як long double
- `const long double M = 2.58316l;` // зберігається як long double
- Визначення константи за допомогою директиви #define :
- `#define PI 3.14159`

● **Вирази**

- **Вирази** - об'єднання операцій і операндів.

● **Приклад .**

- `int b, a = 10;`
- `b = a + 25;`
- Знак "=" - це знак присвоєння, а не знак рівності.

- **Типи операцій :**

- арифметичні;
- відношень;
- логічні і порозрядні;
- інкремента і декремента;
- присвоєння.

- **Арифметичні операції:**

- * - множення; / - ділення; + - додавання; - - віднімання;
- % - ділення по модулю (узяття залишку від цілочисельного ділення).

- **Операції відношень :**

- < - менше;
- <= - менше або рівно;
- > - більше;
- >= - більше або рівно; == - рівно; != - не рівно.

● Логічні операції

- && - логічне «І»;
- || - логічне «АБО»;
- ! - логічне заперечення.

● Властивості логічних операцій

v1	v2	!v2	v1 && v2	v1 v2
false	false	true	false	false
false	true	false	false	true
true	false		false	true
true	true		true	true

● Властивості порозрядних операцій

x	y	~y	x & y	x y	x^y
0	0	1	0	0	0
0	1	0	0	1	1
1	0		0	1	1
1	1		1	1	0

Операція інкремент і декремент :

++ – інкремент; -- – декремент.

оператор $j = i++$; (постфіксна форма запису) еквівалентний $j = i$; $i++$;

оператор $j = ++i$; (префіксна форма запису) еквівалентний $i++$; $j = i$.

Приклад.

```
#include "stdafx.h"
#include <stdlib.h>
#include <iostream>
#include <conio.h>
using namespace std;
int _tmain(int arge, _TCHAR* argv[])
{   int sum, a=2, b=5;
    sum = a + b++;
    cout<<a<<' '<<b<<' '<<sum<<endl;    // 2 6 7
    a=2; b=5;
    sum = a+++b;
    cout<<a<<' '<<b<<' '<<sum<<endl;    // 3 5 7
    a=2; b=5; sum = (a++)+b;
```

Приклад (продовження).

```
cout<<a<<' '<<b<<' '<<sum<<endl; // 3 5 7
a=2; b=5; sum = a(++b);
cout<<a<<' '<<b<<' '<<sum<<endl; // 2 6 8
_getch(); }
```

- ***Операції з присвоєнням:***
- **=, /=, +=, -=, %=, <<=, >>=, &=, ^=, |= .*
- ***Приклад .***
- *cnt+=50; // cnt=cnt+50;*
- *cnt%=2; // cnt=cnt%2;*

Перетворення типів

Правила приведення типів:

1. Автоматично здійснюються тільки ті перетворення, які перетворюють операнди з меншим діапазоном значень в операнди з більшим діапазоном значень, наприклад:

```
int i_var = 5;  
float f_var = 2.5, summa;
```

...

```
summa = i_var + f_var;
```

2. Вирази, що не мають змісту (наприклад, число з плаваючою комою в ролі індексу), не пропускаються компілятором ще на етапі трансляції:

```
float f;
```

...

```
mas [f]=25; // викликає помилку трансляції (Error)
```

3. Вирази, в яких могла б втрачатися інформація (наприклад, при присвоєнні довгих цілих коротшим або дійсних цілим), можуть викликати попередження (Warning), але вони допустимі:

```
int i;
```

```
float f=3.2;
```

```
i=f; // попередження (Warning) при трансляції
```

Маніпулятор setw

- **Маніпулятор setw** дозволяє задати ширину поля виведення даних. Ширина визначає кількість знакомісць (символів), які відводяться під дані, що виводяться. Для його підключення потрібний заголовний файл <iomanip>.
- **Приклад.**
- #include "stdafx.h"
- #include <stdlib.h>
- #include <iostream>
- #include <conio.h>
- #include <iomanip>
- using namespace std;
- int _tmain(int arge, _TCHAR* argv[])
- { long IVar = 5312647;
- int iVar = 536;
- char cVar = '*';
- cout << setw(10) << IVar << setw(5) << iVar
- << setw(3) << cVar<< endl;
- _getch();}

ЛІНІЙНІ І РОЗГАЛУЖЕНІ АЛГОРИТМИ

- **Лінійні алгоритми**
- Лінійними називаються програми, в яких оператори виконуються один за одним від першого до останнього, не повторюючись і не змінюючи порядку свого виконання.

- **Основні математичні функції мови C++**

Функція	Позначення функції	Тun		Ім'я файлу опису
		значення функції, що повертається	аргумент	
Абсолютне значення	abs(x)	int	int	<stdlib.h>
	cabs(x)	double	double	<math.h>
	fabs(x)	float	float	<math.h>
Арккосинус	acos(x)	double	double	<math.h>
Арксинус	asin(x)	double	double	<math.h>
Арктангенс	atan(x)	double	double	<math.h>
Косинус	cos(x)	double	double	<math.h>
Синус	sin(x)	double	double	<math.h>
Експонента e^x	exp(x)	double	double	<math.h>
Статечна функція x^y	pow(x, y)	double	double	<math.h>
Логарифм натуральний	log(x)	double	double	<math.h>
Логарифм десятковий	log10(x)	double	double	<math.h>
Корінь квадратний	sqrt(x)	double	double	<math.h>
Тангенс	tan(x)	double	double	<math.h>

● **Приклад 3.3. Змінити місцями значення змінних x і y.**

- `#include "stdafx.h"`
- `#include <iostream>`
- `#include <math.h>`
- `#include <conio.h>`
- `using namespace std;`
- `int _tmain(int argc, _TCHAR* argv[])`
- `{ setlocale(LC_ALL, "Russian");`
- `int x, y, wrk; // wrk - робоча змінна`
- `cout << "Введіть x і y ->"; cin >> x >> y;`
- `cout <<"x = " << x << " y = " << y << endl;`
- `wrk = x; // запам'ятати в wrk значення змінної x`
- `x = y; // помістити в x значення y`
- `y = wrk; // помістити в y значення x, що зберігається в wrk`
- `cout <<"x = " << x << " y = " << y << endl;`
- `_getch();`
- `return 0; }`

Функції форматного введення-виведення даних

printf ("управляючий_рядок", [список_аргументів])

Список аргументів - це послідовність констант, змінних або виразів, значення яких виводяться на екран дисплея відповідно до формату управляючого рядка. Список аргументів у функції printf() може бути відсутнім.

Управляючий рядок визначає кількість і тип аргументів. Містить об'єкти трьох типів : звичайні символи, що виводяться на екран без змін, специфікації перетворення, кожна з яких викликає виведення на екран значення чергового аргументу зі списку аргументів, і символічні управляючі константи. Кожна специфікація перетворення починається з символу % і закінчується символом перетворення.

Між ними можуть записуватися:

- знак мінус (-), що вказує на те, що текст, який виводиться, вирівнюється по лівому краю; за замовчуванням вирівнювання відбувається по правому краю;

- рядок цифр, що задає мінімальний розмір поля виведення;

- крапка, що є роздільником;

- рядок цифр, що задає точність виведення;

- символ l, вказує на те, що відповідний аргумент має тип long.

Символи перетворення :

- d - аргумент перетворюється в десяткове представлення;
- o - аргумент перетворюється у вісімкове представлення;
- x - аргумент перетворюється в шістнадцятиричне представлення;
- c - значенням аргументу є символ;
- s - значенням аргументу є рядок символів;
- e - значенням аргументу є величина типу float або double в експоненціальній формі запису;
- f - значенням аргументу є величина типу float або double у формі запису з десятковою точкою;
- g - один з форматів f або e;
- u - значенням аргументу є ціле беззнакове число;
- p – значенням аргументу є покажчик (адреса).

Функція *scanf()*:

***scanf* ("управляюча строка", список_аргументів);**

Список аргументів є обов'язковим.

Аргументи функції *scanf()* повинні бути покажчиками на відповідні значення, для цього перед ім'ям змінної записується символ **&**. Управляючий, рядок містить специфікації перетворення і використовується для визначення кількості і типів аргументів.

Приклад:

```
printf ("i=%d,\n j=%d,a=%6.2f.\n",i,j,a);
```

Якщо i= 1234, j=127, a=86.531, то на екрані маємо:

i =1234,

j =127, a = 86.53.

Допустиме використання символів заповнення:

```
printf ("i =%06d,\n j=%d, a=%6.2f.\n", i, j, a);
```

Значення і виглядає так:

i =001234

```
scanf ("%d %f %c %s", &i,&a,&ch, r);
```

Розгалужені алгоритми

- Передбачають вибір маршруту виконання програми залежно від істинності або помилковості деяких умов.
- **Конструкції прийняття рішення :**
- if
- if - else
- switch.

- **Умовний оператор if**

- ***if (вираз){оператор1***
- ***оператор2;***
- ***операторN; } //end if***



- if (выраз == значения) оператор;
- if (выраз!=значення) оператор;
- if (выраз>значення) оператор;
- if (выраз!=0) оператор;
- if (выраз) оператор;
- if (вираз = значення) оператор; **// ПОМИЛКА**

- **Приклад.**

- `#include "stdafx.h"`
- `#include <iostream>`
- `#include <conio.h>`
- `using namespace std;`
- `int _tmain(int argc, _TCHAR* argv[])`
- `{ setlocale(LC_ALL, "Russian");`
- `int y, c;`
- `cout << "Введіть рік:";`
- `cin >> y;`
- `c = y/100;`
- `if (y%100 != 0)c+=1;`
- `cout << «Цей рік належить століттю»<< c << endl;`
- `_getch();`
- `return 0; }`

● Оператор *if- else*

- *if* (вираз){ оператор1;
- оператор2 }
- *else* {оператор3;
- оператор4;}



• Вкладені галуження:

- *if* (выраз1) оператор1;
- *else if* (выраз2) оператор2;
- *else if* (выраз3) оператор3;
- *else if* (выразN) операторN;
- *else* // необов'язкова частина
- оператор_за замовчуванням;

- **Умовний оператор :**
- *змінна = вираз? значення1: значення2;*
- **Умовний вираз :**
- *if (вираз) змінна = значення1;*
- *else змінна = значення2;*

<pre>if (test == 'Y') TestValue = 100; else TestValue = 0;</pre>	<pre>... TestValue = (test == 'Y')?100:0; ...</pre>
--	---

- **Оператор множинного вибору**
- *switch (вираз)*
- *{case значення1: оператор1;*
- *break;*
- *case значення2: оператор2;*
- *break;*
- *case значення3: оператор3;*
- *break;*
- *default: // необов'язковий компонент*
- *оператор_по_непорівнянню; // якщо не було жодного*
- *// співпадання.*
- *} //end switch (вираз)*

- **Приклад.**
- #include "stdafx.h"
- #include <iostream>
- #include <conio.h>
- using namespace std;
- int _tmain(int arge, TCHAR* argv[])
- { setlocale(LC_ALL, "Russian");
- char sym;
- int op1, op2, res;
- cout << "Введите символ арифметической операции ";
- cin >> sym;
- switch (sym)
- {case '+': cout << "Сложение" << endl;
- cout << "1 слагаемое:";
- cin >> op1;
- cout << "2 слагаемое:";
- cin >> op2;
- res = op1 + op2;
- cout << op1 << '+' << op2 << '=' << res << endl;
- break;

- case '-':

```
cout << "Вычитание" << endl;
cout<<"Уменьшаемое:";
cin >> op1;
cout<< "Вычитаемое:";
cin >> op2;
res = op1 - op2;
cout << op1 << '-' << op2 << '=' << res << endl;
break;
```
- case '*':

```
cout << "Умножение" << endl;
cout<< "Множимое:";
cin >> op1;
cout<<"Множитель:";
cin >> op2;
res = op1 * op2;
cout << op1 << '*' << op2 << '=' << res << endl;
break;
```
- case '/':

```
cout << "Деление" << endl;
cout<<"Делимое:";
cin >> op1;
cout<<"Делитель:";
cin >> op2;
```


- if (op2 != 0)
- {res = op1 / op2;
 cout << op1 << '/' << op2 << '=' << res << endl;}
 else cout <<"Деление на 0 запрещено" <<
 endl;
 break;
- default:cout<<"Не арифметическая операция"<< endl;
- } // end switch (sym)
- _getch();
- return 0; }

- ***Визначити значення змінної w після виконання наступних операторів :***

- w = 100; u = 30;
- switch (u/7)
- { case 0: w = 0; break;
- case 1: w = 1; break;
- case 2: w = 2; break;
- case 3: w = 3; break;
- default: w = 7; }

Визначити значення змінної m після виконання наступних операторів :

$m=5;$

$\text{if } (x>0) \{ \text{if } (y>0) m = 10; \}$

$\text{else } m = 20;$

а) при $x = -5, y = 7;$

б) при $x = 2, y = -3;$

в) при $x = 9, y = 3.$